Copy No. 3

7N-81-CR
210980
21P

INSTITUTE FOR COMPUTER APPLICATIONS IN

SCIENCE AND ENGINEERING (ICASE)

TRACE DRIVEN STUDIES OF DEADLOCK CONTROL AND JOB SCHEDULING

Stephen W. Sherman

John H. Howard, Jr.

and

James C. Browne

(NASA-CR-180459) TRACE DRIVEN STUDIES OF DEADLOCK CONTROL AND JOB SCHEDULING (ICASE) 21 p

N89-70880

Unclas

00/81 0210980

TRACE DRIVEN STUDIES OF DEADLOCK

CONTROL AND JOB SCHEDULING

Stephen W. Sherman
Institute for Computer Applications in Science and Engineering
(On leave from the University of Houston)

John E. Howard, Jr., and James C. Browne
The University of Texas at Austin

APPROVED BY: ______________________
James M. Ortega, Director
Institute for Computer Applications
in Science and Engineering

This research was supported by the National Science Foundation under grant GJ-1084, "Design and Analysis of Operating Systems" to the University of Texas at Austin and was made possible by the cooperation of the Computation Center of the University of Texas at Austin. The work was completed while the first author was in residence at ICASE, which is supported by NASA Grant NGR 47-102-001 at Langley Research Center.

Abstract

A trace-driven model is used to study the effects of various schedulers and deadlock control algorithms in a general - purpose operating system. Jobs' requests for resources are extracted from a production load and used to drive a detailed simulation program. The simulation results show that the preemptive deadlock control algorithms give consistently good performance in terms of CPU utilization. The bankers algorithm and the detection and recovery deadlock control algorithms are susceptible to "knotting" (holding of resources by a blocked process) when there is no preemption, but their performance can be improved significantly by 1) allowing a moderate amount of preemption and 2) by forcing the job scheduler to limit the number of jobs competing for resources. When "knotting" is limited by either of the above methods, non-preemptive job scheduling improves CPU utilization. This paper extends and develops previous work and summarizes the interaction between some characteristics of job schedulers and deadlock control algorithms.

I. Introduction

A variety of algorithms for dealing with the deadlock problem in operating systems have been proposed and compared qualitatively ([1] [2] [3] [4]). This paper reports a quantitative study of the effects of deadlock control algorithms and job schedulers on CPU utilization.

CPU utilization is compared for two job loads of equal resource requirements. The actual job load had a number of interactive jobs which are reflected in an interactive model. Results from experiments with the interactive model are compared to results from a batch model which uses the same jobs as the interactive model but treats all jobs as if they were batch jobs.

Trace-driven modeling ([5] [6] [7]) is the vehicle used for this study. It is a simulation technique based on a detailed job load extracted from a production system, and is completely and historically described in [6]. It has also been used to study other system algorithms such as CPU scheduling ([5] [7]). A preliminary study of deadlock control algorithms in a batch environment, considering only the effect on CPU utilization, appears in [8]. The more sensitive measure of response time for interactive jobs was considered in [9].

This paper updates results from [8] and combines previous work ([5] [9]) to develop new results on the positive effects of preemption on certain deadlock control algorithms. Further experiments are also presented which support conclusions in [9] concerning the need for an intelligent job scheduler preceding the deadlock control algorithm and the general good performance of preemptive deadlock control algorithms. The details of experimental precedures used to obtain the results reported in this paper have been previously reported in several papers ([5][6][7][8][9]) and are therefore only briefly sketched in this paper.

II. The Model and Environment

Trace-driven modeling is a technique whereby a recorded trace of system activities is directly used to define the environment and workload for a model of a computer system ([6]). Trace-driven modeling is a form of simulation which can be accurately validated. The model is validated by comparing its performance with the performance of the system whose data was recorded. Further information of trace-driven modeling in general can be found in [6] and very detailed information on this particular trace-driven modeling effort is in [5].

A CDC 6600 ([10]) was used to gather the trace data. The locally written operating system UT-2 ([5] [11] [12]) can support up to 13 user jobs and 3 system jobs concurrently. The user jobs have five peripheral processors for input/output, swapping, and system control functions.

The UT-2 system is a multiprogramming system that supports a mixed batch (5000 to 6000 jobs per day) and interactive (35 to 45 users simultaneously) load. Measurements used in the model were taken over a relatively long 30 minute period and a shorter 3 minute period with 1400 and 220 interactions respectively from the interactive users.

Two models are used to study the system. The first and earliest model ([5][8]) treats all jobs in the system as batch jobs. This model will be referred to as the batch model. The interactive jobs are included in this model in terms of their resource requirements, but they are not given any special priority and are not preempted for think time. That is, think time is assumed to be zero. The batch model was a reasonable model for the UT-2 system at the time the measurements were taken. Early versions of the system were completely batch oriented. A large majority of the users only used the batch system. The interactive system had only batch versions of language processors and utilities available to it. Figure 1 (without the interaction complete path) is an illustration of job processing in the batch model.

When the interaction complete path is included, Figure 1 shows a model of job processing in the interactive model. The interactive model is similar to the batch model in all respects except: 1) Interactive jobs are swapped out while waiting for input. 2) Think times are taken from the trace data and 3) The job scheduler treats interactive jobs in a slightly different way (explained below) than batch jobs. More detailed information on the interactive model may be found in [9].

The two job schedulers examined in the models are the preemptive scheduler (SP) that preempted jobs whenever less expensive jobs arrived in the job queue and a non-preemptive scheduler (SNP) which would not preempt at all in the batch model and limited preemption to interactive jobs in the interactive model. The actual scheduler used in the UT-2 system is very similar to the preemptive scheduler ([11]). Both the preemptive and non-preemptive schedulers had the following characteristics: 1) A cost was assigned to each job equal to the product of its current memory requirements and the amount of CPU time it needed before completing its current transaction. 2) The jobs were sorted in order of increasing cost. 3) The jobs were scanned least - cost first and any job that would fit into the available memory was selected. 4) At most 4 interactive jobs could be selected.

As in previous studies([5][8][9]) four deadlock control algorithms are studied. The resources considered by the deadlock algorithms are central memory and peripheral processors. Immediate preemption, IP, is the technique used in the UT-2 system. If a job's request for memory cannot be satisfied immediately, the job is swapped out. Complete assignment, CA, prevents deadlocks by initially assigning to a job all of the resources it will ever need. Detection and recovery ([3]), DR, consists of running a deadlock detection algorithm whenever a job's request for additional resources cannot be satisfied and recovering if deadlock is detected. The "bankers algorithm" ([4]), BA, avoids deadlocks by assigning resources only when the system can find at least one safe sequence in which it can run all jobs.

III. Validation

In order not to distort the load presented to the simulated job scheduler ([6][9]), an initial queue of 55 jobs was selected from the pool of known jobs. The total amount of processing time used by the initial queue is stored as a threshold. Whenever the remaining processing in the simulated input queue drops below this threshold, new jobs are selected from the unused jobs.

Table 1 presents validation information for the relatively long measurement period. The first four columns represent data used in [9]. The actual and simulated interactive system performance measures (first two columns) agree to within 3.5 °/o, with the measure most important here (CPU utilization) in agreement to within 1 °/o relative error. The third column displays simulation results with the overhead associated with the software event recorder removed, showing that its effect was a degradation by approximately 2 °/o. The fourth column gives results from a simulation with a different random ordering of the jobs, and again displays relative deviations of about 2 °/o. The fifth column through the seventh column present the simulated batch system performance measures ([5]). Columns five and six correspond well to the simulated interactive system performance measures in columns two and three showing a slight improvement in performance due to fewer preemptions resulting in less overhead. The random ordering of the jobs in column seven was different from the ordering reflected in the results of the fourth column. The processing threshold for the initial set of jobs was about 11 °/o of the threshold used in the initial ordering. This meant that the permuted batch model had a smaller number of jobs to consider throughout the run, and therefore the permuted batch model generated a lower degree of multiprogramming and utilized less memory. Even with this unfortunate random choice of the initial set of jobs, the model still agreed with the actual system in CPU utilization to within 2.5 °/o. We conclude from this information that we have constructed a valid and stable simulation model of the actual system.

The short measurement period contains too few interactions and jobs to allow a convincing validation. Table 2 gives the comparison between actual and simulated system performance. The results based on the short period must be taken as showing trends only and as lending credence to the validated data from the longer measurement period.

IV. Results

Table 3 gives CPU utilizations for each combination of schedulers (SP and SNP), deadlock control algorithms (IP, CA, DR and BA) and models (interactive and batch) for the long measurement period. Table 4 gives the same set of CPU utilizations for the short measurement period.

In the interactive model in Table 3, the performance of each deadlock algorithm improved from 3.03% to 8.66% when the non-preemptive scheduler replaced the preemptive scheduler. The improvement in CPU utilization coincides with the decrease in overhead and delays caused by swapping. However, it is important to note that the non-preemptive scheduler, SNP, still swaps jobs in the interactive model due to the requirements for reasonable response time. (A parallel study ([9]) has shown that response times are worse using the SNP scheduler with the interactive model than with the SP scheduler). The batch model in Table 3 shows a performance improvement similar to the improvement in the interactive model using the IP and CA deadlock control algorithms with the SNP scheduler. However, the non-preemptive scheduler yields less CPU utilization than the preemptive scheduler using the DR and BA deadlock algorithms. This loss of CPU utilization for the detection and recovery algorithm (6.01%) and the bankers algorithm (18.40%) is not reflected in the batch model for the short measurement period in Table 4.

The significant characteristic which appeared in the DR and BA deadlock algorithms with the non-preemptive scheduler was the appearance of the "knotting" phenomenon. Knotting is the degradation of performance brought about by the tying-up of resources by jobs which are not able to make effective progress ([8]). The deadlock algorithm DR and BA consider memory and peripheral processors as their non-preemptable resources and the number of peripheral processors being held by jobs requesting central memory is an indication of knotting. Table 5 shows that as the number of jobs holding peripheral processors and waiting for central memory increases, the CPU utilization decreases.

In the experiments in Table 3, the interactive model seems to have too much preemption activity using the preemptive scheduler. The performance was improved by using a non-preemptive scheduler and only preempting for a limited number (a maximum of 4 at one time) of interactive jobs. The decrease in preemption overhead overcame any tendency for the system to knot using the DR and BA algorithms. The performance of the batch model using the preemptive scheduler was almost equivalent to the interactive model. The non-preemptive scheduler was strictly non-preemptive in the batch environment and the lack of job preemption allowed knotting to dominate the resource environment for the DR and BA deadlock algorithms and more than compensate for any gains due to reduced overhead that were apparent in the IP and CA deadlock algorithms. Only a modest amount of preemption seemed to be needed to deter knotting since the interactive jobs averaged less than 1 interaction per second during the long measurement period. Statistical techniques ([5]) used to analyze the contributions of the deadlock control algorithms (IP, CA, DR, BA), job schedulers (SP, SNP), and models (batch, interactive) show that almost 60 °/o of the observed variations in CPU utilization are accounted for by the interactions between variables. This large cross term indicates the significance of those interactions in designing an operating system. The need for preemption in the DR and BA algorithms is very unfortunate since these algorithms are usually used when preemption is difficult to achieve. The short measurement period in Table 4 was too short for knotting to develop. Jobs finished and resources were freed at a very rapid rate. Therefore, the non-preemptive scheduler always performed better than the preemptive scheduler.

In the batch model, some perturbations of the bankers algorithm and the detection and recovery algorithm were tried using the non-preemptive scheduler in an attempt to achieve better performance. Knotting had led to deadlocks in the detection and recovery algorithm. While only one deadlock was detected using the premptive scheduler, 14 deadlocks occurred with the non-preemptive scheduler. The recovery procedure when a deadlock was detected was to preempt the resources currently held by the last job that caused the deadlock. When the recovery procedure was changed to preempt all of the jobs that held resources contributing to the deadlock, the CPU utilization increased from 61.40% to 67.45% using the non-preemptive scheduler.

An alternative technique for the bankers algorithm consisted of treating the peripheral processors requesting memory as preemptable resources. The batch model was changed to place the peripheral processor program that requested memory at the end of the queue of peripheral processor programs. The new technique returned an increase in CPU utilization of 9.02% over the CPU utilization of 48.15 °/o presented in Table 3 for the non-preemptive scheduler.

In an attempt to reduce the contention for resources, the maximum number of user jobs is reduced in steps of 2 from 13 to 3 jobs. Three system jobs are always active. Table 6 shows the results of this experiment. The bankers algorithm steadily increased its performance under the non-preemptive scheduler until only a maximum of 3 user jobs were allowed. As a comparison, the same experiment was run using the preemptive scheduler with the immediate preemption deadlock control algorithm. The results of that experiment show little change in the CPU utilization with a barely preceptable downward trend when a maximum of 5 user jobs was allowed. Clearly,intelligent scheduling that removes congestion will aid the bankers algorithm. Similar results were found in [9].

The CPU utilizations of DR and BA reported here are markedly superior to those reported in the preliminary study ([8]). This is due both to the use of a model that is more comprehensive in its resolution of job characteristics and to the correction of an invalid implementation of DR and BA. Precise information on resource requirements, available and utilized in this trace-driven model, is highly favorable to the performance of CA and BA. Such precise information on resource requirements is not often available in normal production environments. The cost of preempting jobs on the system modeled in this study is very small since preempted jobs are swapped to extended core storage. The ease of preemption certainly helps the IP deadlock algorithm. The batch model was changed to assess the system a penalty in CPU time whenever a job was preempted in order to make preemption more expensive.

Two experiments were conducted with system penalties of 100 milliseconds and 1 second using the SNP scheduler and the IP deadlock algorithm. CPU utilization dropped from 77.98 °/o to 76.58°/o for the 100 millisecond penalty and to 60.29°/o for the 1 second penalty. One second of CPU time is a tremendous amount on a CDC 6600 and the performance of the IP deadlock mechanism under a penalty situation indicates that preemption should certainly be considered for deadlock control even if the cost is very high.

V. Conclusions

The simulation results presented here support the following conclusions. Non-preemptive job schedulers combined with the immediate preemption and complete assignment deadlock control algorithms yield better performance in terms of CPU utilization than preemptive job schedulers. The detection and recovery algorithm and the bankers algorithm are very susceptible to knotting when no preemption is allowed. A moderate amount of preemption can greatly improve the performance of both the detection and recovery algorithm and the bankers algorithm. The performance of deadlock control algorithms that are subject to knotting can also be improved by limiting the number of jobs competing for resources. CPU utilization can be improved in the detection and recovery algorithm by preempting all of the jobs that cause a deadlock rather than preempting the minimum number of jobs. The preemptive deadlock control algorithm gave consistently good performance even when a penalty for preemption was accessed.

References

1. Coffman, E. G., Elphick, M., and Shoshani, A., System deadlocks. Computing Surveys 3, 2 (June 1971), 67.

2. Holt, R. C., On deadlock in computer systems. Ph.D. dissertation, Department of Computer Science, Cornell University, Ithaca, N.Y., January 1971.

3. Shoshani, A., and Coffman, E. G. Detection and prevention of deadlocks. Fourth Annual Princeton Conf. on Information Sciences and Systems, Princeton, N. J., March 1970.

4. Habermann, A. N., Prevention of system deadlocks. Comm. ACM 12, 7 (July 1969), 373.

5. Sherman, S. W., Trace-driven modeling studies of the performance of computer systems. Ph.D. dissertation, Department of Computer Sciences, University of Texas, Austin, Texas, 1972.

6. Sherman, S. W., and Browne, J. C., Trace-driven modeling: Review and overview. **Symposium on the Simulation of Computer Systems**, Gaithersburg, Md., June 1973.

7. Sherman, S. W., Baskett, F., and Browne, J. C., Trace-driven modeling and analysis of CPU scheduling in a multi-programming system. Comm. ACM 15, 12 (Dec. 1972), 1063.

8. Sherman, S. W., Howard, J. H., and Browne, J. C., A comparison of deadlock prevention schemes using a trace-driven model. Sixth Princeton Conf. on Information Sciences and Systems, Princeton, N.J., March 1972, p. 604.

9. Sherman, S. W., Howard, J. H., and Browne, J. C., A study of response times under various deadlock algorithms and job schedulers. 1974 ACM National Conf., San Diego, Cal.

10. Thornton, J. E., Design of a Computer: The CDC 6600. Scott, Foresman and Co., Glenview, Ill., 1970.

11. Howard, J. H., A large-scale dual operating system. 1973 ACM National Conf., Atlanta, Ga., p. 242

12. Johnson, D. S., A process-oriented model of resource demands in large multiprocessing computer utilities. Ph.D. dissertation, Department of Computer Sciences, The University of Texas, Austin, Texas, 1972.

Figure 1. System Model

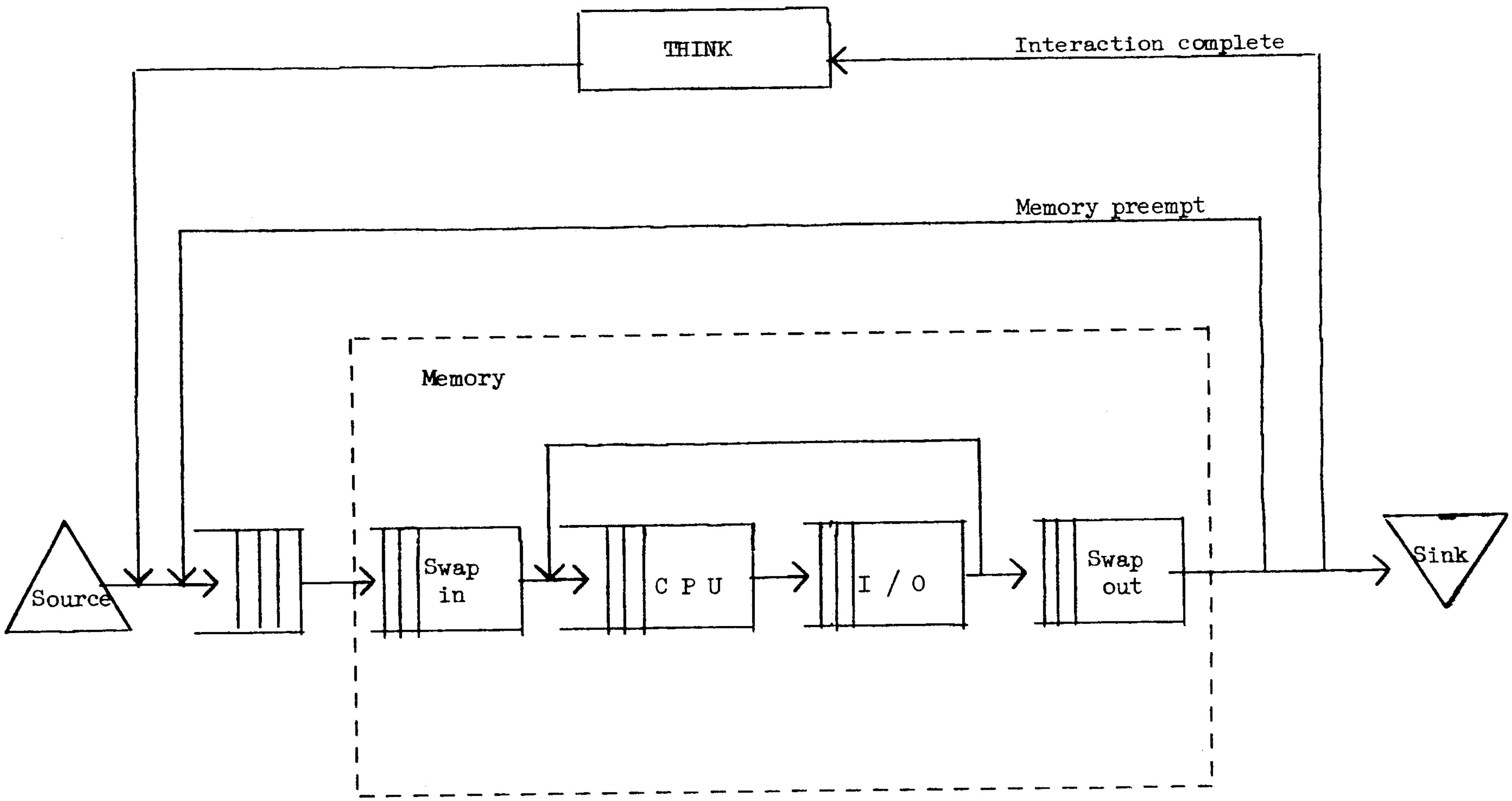

Table 1

VALIDATION OF LONG MEASUREMENT PERIOD

	Actual System	Interactive Model	Interactive Model Without Probe	Interactive Model Permuted	Batch Model	Batch Model Without Probe	Batch Model Permuted
ET[a]	1637.719	1623.747	1578.504	1579.189	1589.34	1555.558	1697.027
CPU[b]	69.73	70.32	72.34	73.30	71.84	73.40	67.28
PPU[c]	82.6	85.5	81.5	83.3	85.9	84.4	79.9
DMP[d]	8.38	8.28	8.33	8.43	8.72	8.76	7.06
Mem[e]	121,880	120,027	115,443	115,346	116,510	116,457	95,617
RT[f]	352	357	336	352			

a) running time in seconds
b) percentage of CPU utilized by user jobs
c) average percent of PPU resources utilized
d) degree of multiprogramming
e) average memory utilized (out of 131,072)
f) response time in milliseconds

Table 2

VALIDATION OF SHORT MEASUREMENT PERIOD

	Actual System	Interactive Model	Interactive Model Without Probe	Batch Model	Batch Model Without Probe
ET[a]	199.172	207.518	202.402	193.353	189.530
CPU[b]	47.16	45.26	46.40	48.58	49.56
PPU[c]	96.7	97.0	94.0	97.8	98.1
DMP[d]	10.17	10.48	10.23	10.79	9.74
Mem[e]	116,930	114,291	109,791	113,194	105,801
RT[f]	1373	1092	842		

Table 3

CPU UTILIZATION (PERCENT)
Long Measurement Period

Interactive Model

	IP	CA	DR	BA
SP	69.32	64.19	67.63	66.81
SNP	77.98	71.30	75.85	69.84

Batch Model

	IP	CA	DR	BA
SP	67.88	65.56	67.41	66.55
SNP	77.98	74.31	61.40	48.15

Table 4

CPU UTILIZATION (PERCENT)
Short Measurement Period

Interactive Model

	IP	CA	DR	BA
SP	42.31	40.83	40.14	38.83
SNP	48.13	45.66	47.03	45.12

Batch Model

	IP	CA	DR	BA
SP	44.57	42.62	43.67	42.80
SNP	50.73	49.33	47.71	45.62

Table 5

The effect of jobs holding peripheral processors and waiting for central memory using scheduler SNP and deadlock algorithms DR and BA.

Long Measurement Period

Number of jobs holding and waiting	Percent of elapsed time using DR	Percent of time CPU was active using DR	Percent of elapsed time using BA	Percent of time CPU was active using BA
0	25.55	75.53	7.20	81.32
1	25.51	66.06	17.73	68.13
2	22.17	59.30	22.74	60.31
3	13.59	54.43	14.36	45.99
4	13.17	35.67	37.95	26.01

Table 6

Reduction of maximum number of user jobs from 13 to 3

		Long Measurement Period					
Maximum number of user jobs		13	11	9	7	5	3
SP scheduler and IP deadlock algorithm	Percent CPU utilization	67.88	67.88	67.88	67.91	67.81	
	DMP *	8.65	8.65	8.62	8.38	7.49	
SNP scheduler and BA deadlock algorithm	Percent CPU utilization	48.15	51.94	54.46	68.97	72.51	63.06
	DMP	10.91	10.67	10.30	9.32	7.86	5.96

* DMP - degree of multiprogramming includes the 3 system jobs